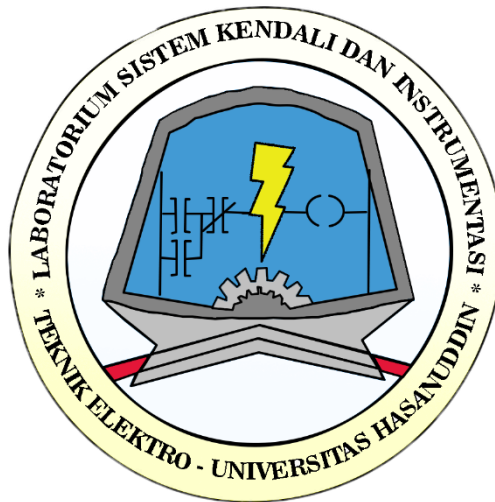


Modul Pembelajaran

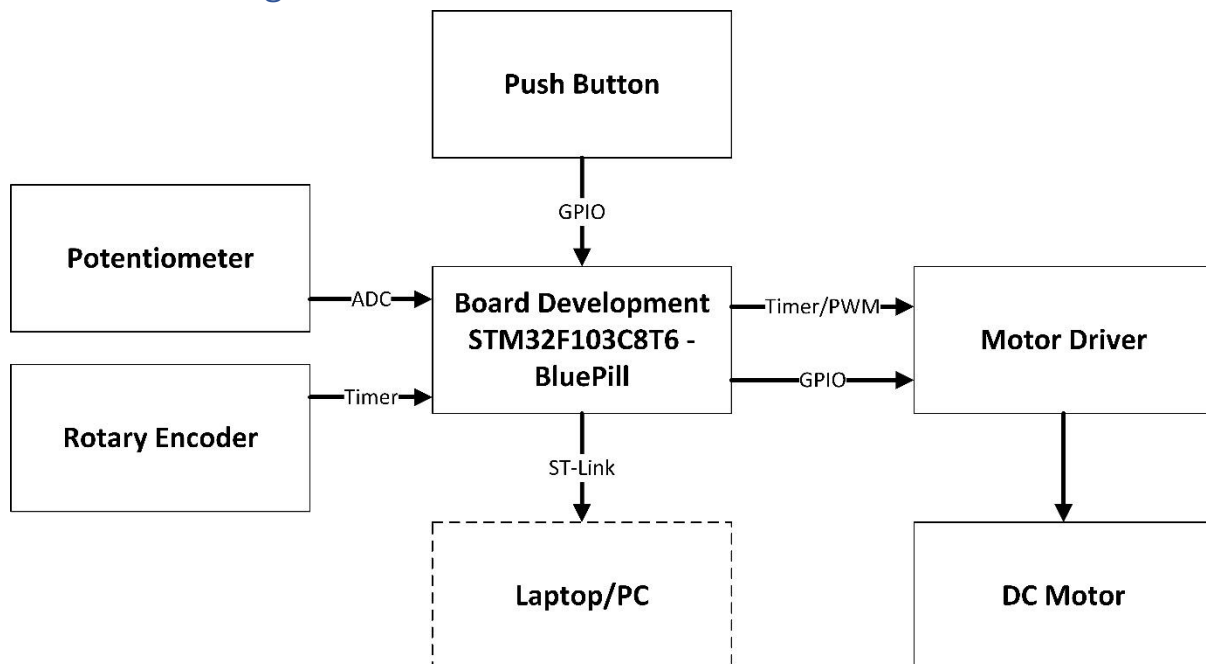


Perancangan Sistem Kendali Berbasis Mikrokontroler STM32

Contents

A. Perancangan Sistem Kendali Motor DC.....	3
B. Petunjuk Struktur Pemrograman	4
C. Setup Project.....	6
D. Pinout Configuration	8
E. GPIO	10
F. Timer-PWM Output	11
G. Timer - Encoder Mode	13
H. ADC.....	15
Latihan Perancangan.....	17

A. Perancangan Sistem Kendali Motor DC



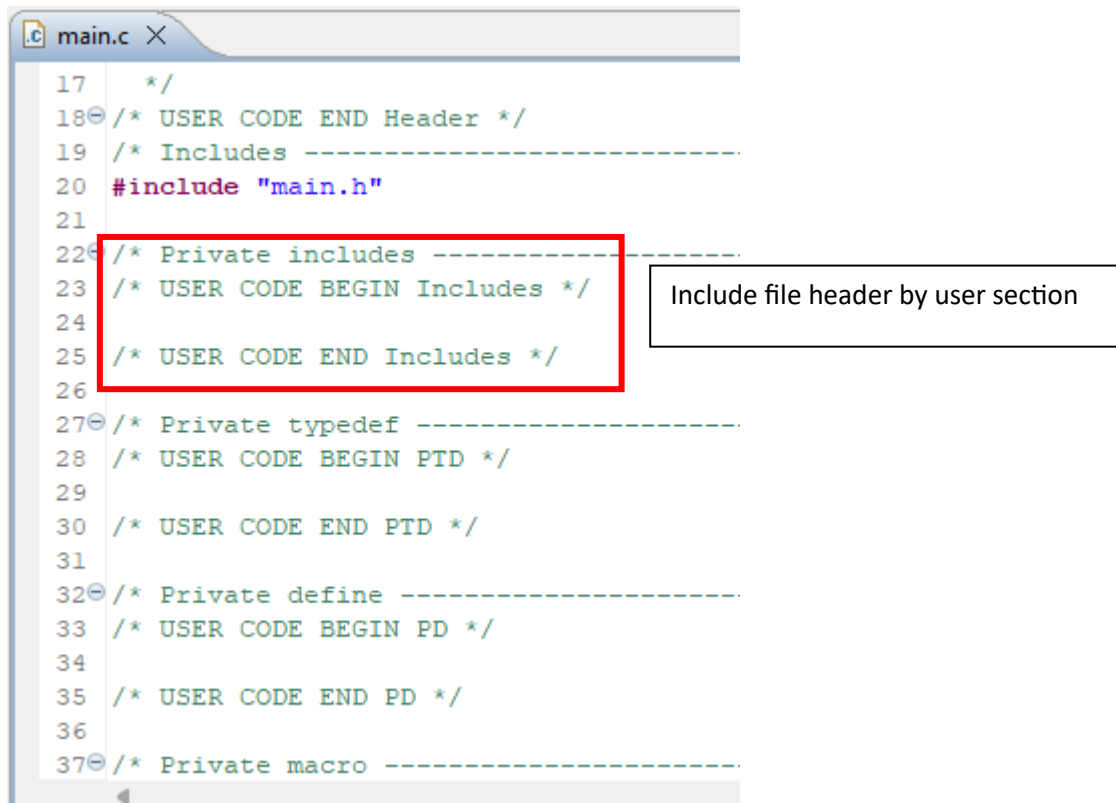
Blok sistem di atas merupakan salah satu contoh perancangan sistem kendali dengan plant motor DC. Masing-masing blok berfungsi sebagai berikut.

1. Push Button: Memberikan input berupa digital (HIGH/LOW) melalui peripheral **GPIO** STM32
2. Potentiometer : Sebagai input berupa analog melalui peripheral **ADC** STM32
3. Motor Driver : Sebagai penghubung antara Motor DC dengan output STM32 yang berupa digital Output (HIGH/LOW) yang melalui peripheral **GPIO** dan **sinyal PWM yang dibangkitkan melalui Timer**.
4. Motor DC : Sebagai Plant pada sistem kendali yang dirancang
5. Rotary Encoder : Sebuah incremental encoder yang tersemat pada motor DC, digunakan untuk mengukur putaran motor DC. Rotary encoder memberikan input berupa pulsa melalui peripheral **Timer sebagai encoder Mode** pada STM32.
6. STM32F103C8T6 : Sebagai pengendali sistem yang dapat disematkan program dari user.
7. Laptop/PC : Digunakan untuk merancang sistem berbasis software STM32CubeIDE, serta dapat melakukan proses debugging program pada hardware STM32

Dari fungsi masing-masing blok, maka pada sistem ini perlu dilakukan konfigurasi pada MCU/MPU STM32F103C8T6 yang mana pinout dari mikrokontroler tersebut diatur menjadi:

1. GPIO Mode Input
2. GPIO Mode Output
3. ADC
4. Timer : PWM Generate
5. Timer : Encoder Mode

B. Petunjuk Struktur Pemrograman



```
17  */
18  /* USER CODE END Header */
19  /* Includes -----
20  #include "main.h"
21
22  /* Private includes -----
23  /* USER CODE BEGIN Includes */
24
25  /* USER CODE END Includes */
26
27  /* Private typedef -----
28  /* USER CODE BEGIN PTD */
29
30  /* USER CODE END PTD */
31
32  /* Private define -----
33  /* USER CODE BEGIN PD */
34
35  /* USER CODE END PD */
36
37  /* Private macro -----
```

Include file header by user section

```

11
42 /* Private variables -----
43
44 /* USER CODE BEGIN PV */
45 int count;
46 /* USER CODE END PV */
47
48 /* Private function prototypes --
49 void SystemClock_Config(void);
50 static void MX_GPIO_Init(void);
51 /* USER CODE BEGIN PFP */
52
53 /* USER CODE END PFP */
54
55 /* Private user code -----
56 /* USER CODE BEGIN 0 */
57 int pushButton;
58 void Fungsi (void){
59
60 }
61 /* USER CODE END 0 */

```

Initiate variable section

Section yang dapat dimanfaatkan user untuk inisiasi variabel dan membuat sebuah fungsi

```

198 MX_GPIO_Init();
199 MX_ADC1_Init();
200 MX_TIM1_Init();
201 MX_TIM2_Init();
202 /* USER CODE BEGIN 2 */
203
204 /* USER CODE END 2 */
205
206 /* Infinite loop */
207 /* USER CODE BEGIN WHILE */
208 while (1)
209 {
210     /* USER CODE END WHILE */
211
212     /* USER CODE BEGIN 3 */
213 }
214 /* USER CODE END 3 */
215 }
216

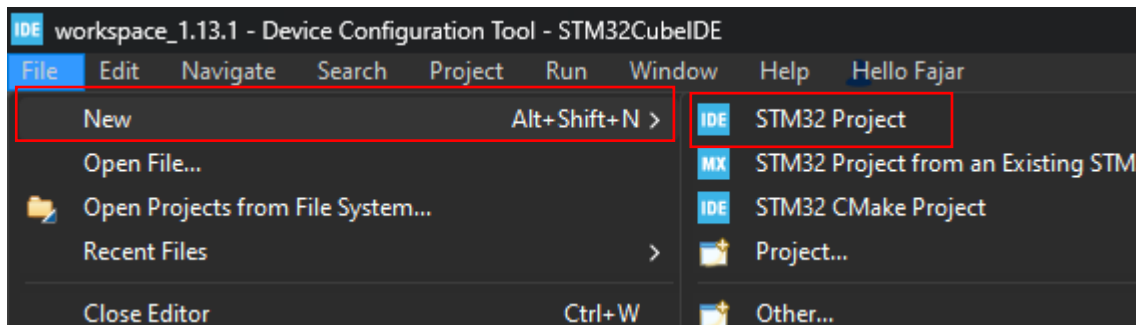
```

Menjalankan fungsi yang dieksekusi sekali saja (seperti void Setup() di arduino)

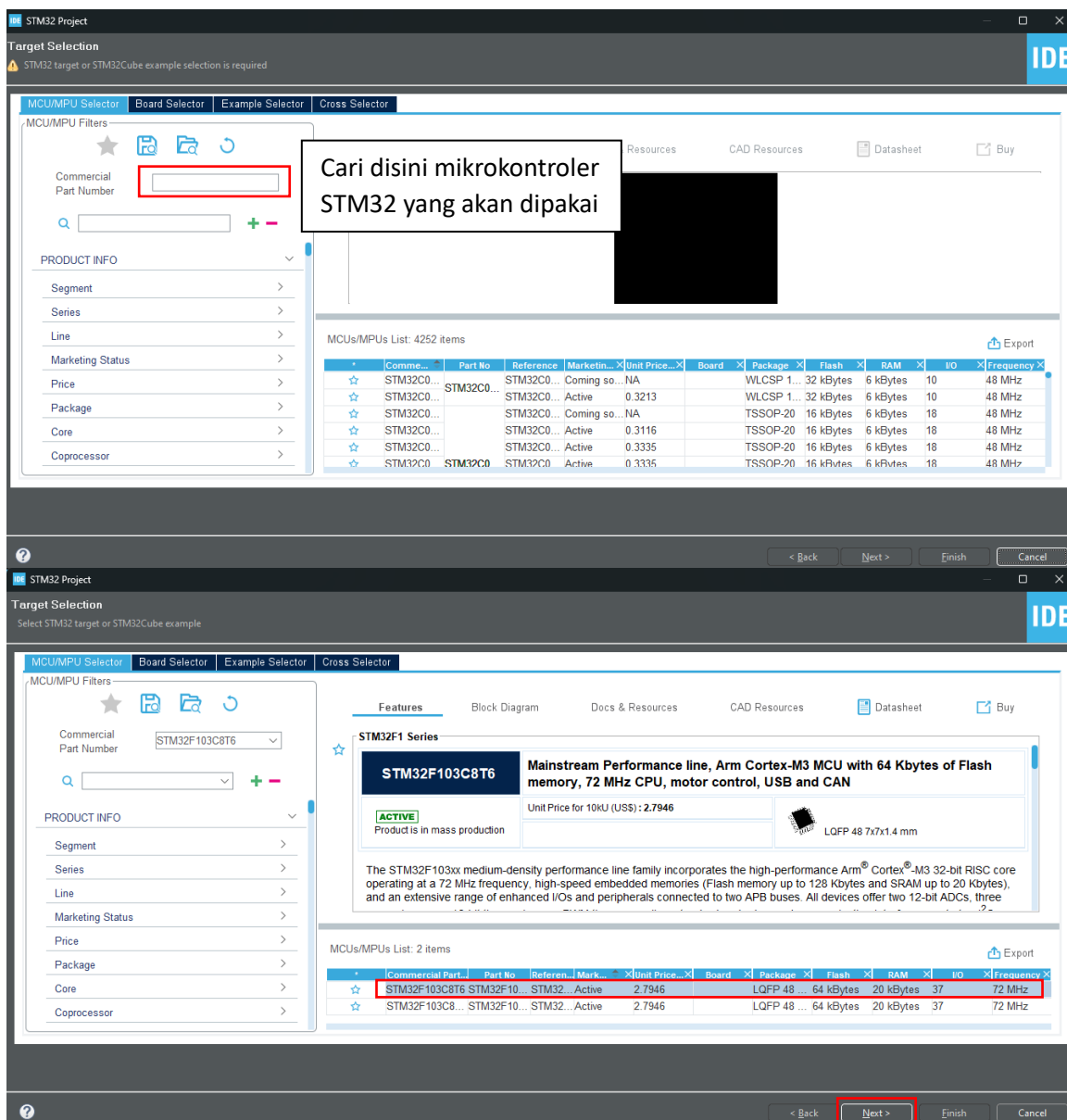
Menjalankan program atau fungsi yg dieksekusi berulang (seperti void Loop() di arduino)

C. Setup Project

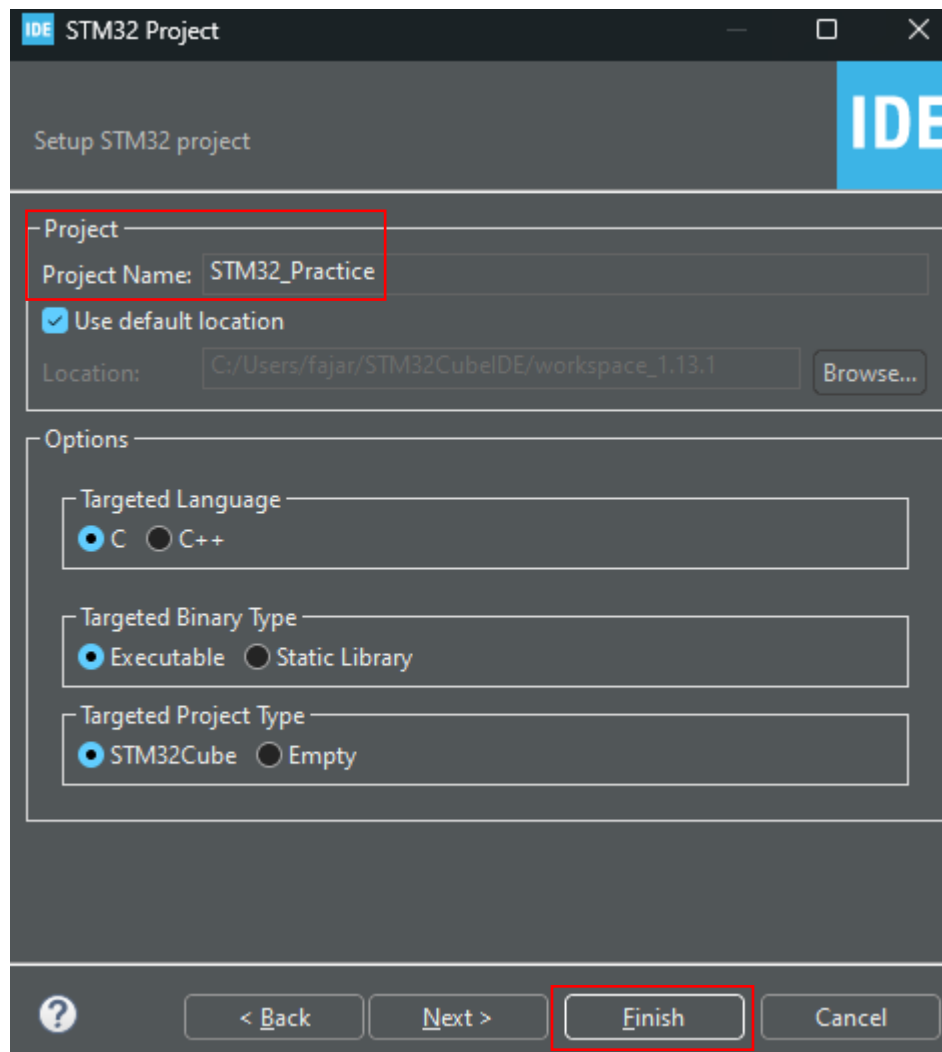
Mula-mula dilakukan setup project untuk memulai project perancangan dengan memilih “New-> STM32 Project” pada Software STM32CubeIDE



Setelah menunggu beberapa saat, pada STM32CubeIDE akan menampilkan CubeFinder untuk mencari MCU/MPU yang akan kita pakai. Dalam hal ini MCU/MPU yang akan digunakan adalah STM32F103C8T6.

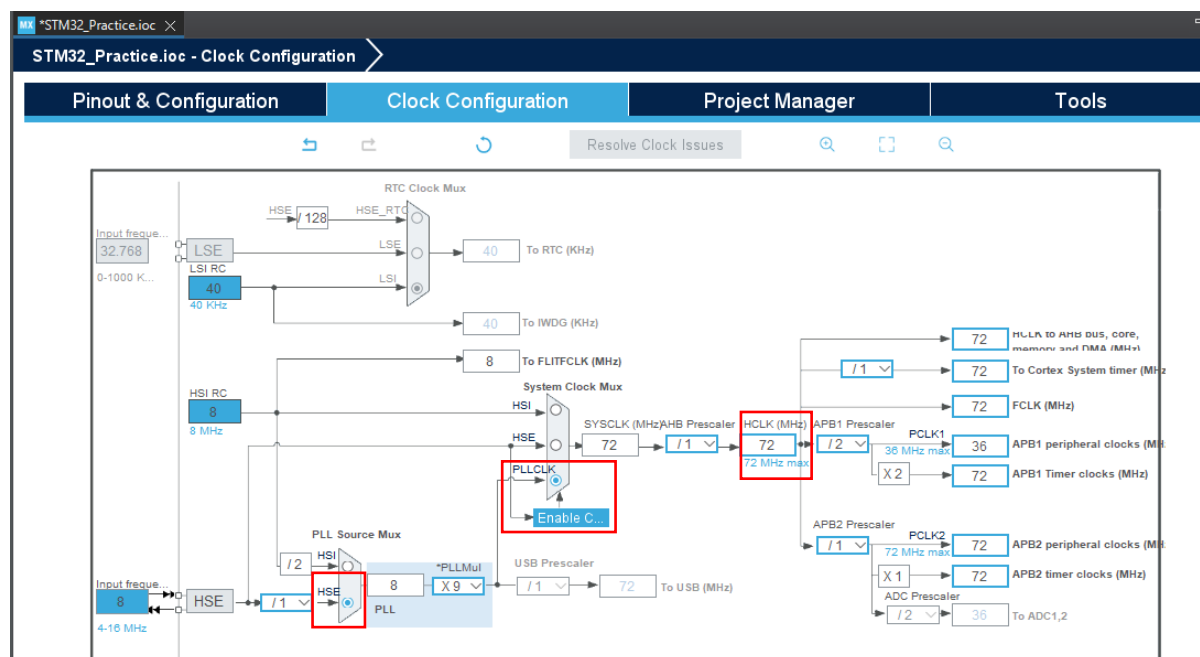
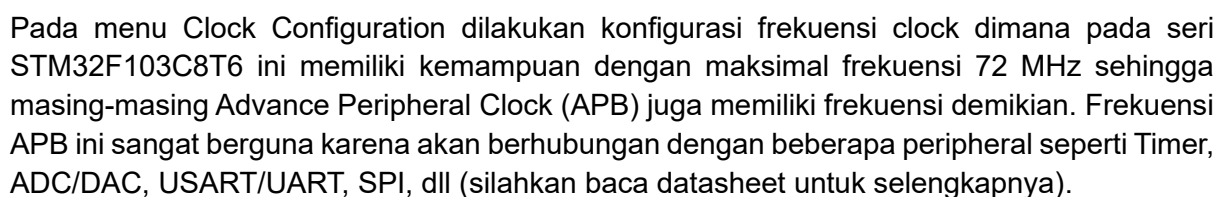


Setelah select MCU STM32 yang akan dipakai, selanjutnya yaitu memberi nama project. Terakhir anda bisa langsung klik next atau langsung klik finish.

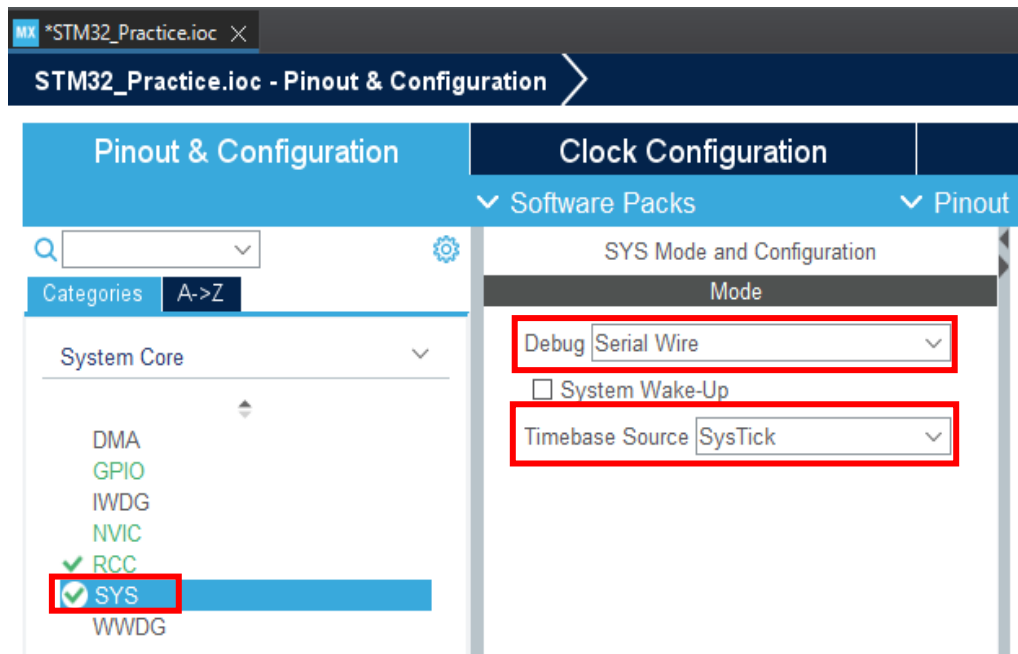


The image shows a screenshot of the 'IDE STM32 Project' setup window. The window has a title bar with 'IDE STM32 Project' and standard window controls. Below the title bar, there's a section titled 'Setup STM32 project'. The main area is divided into sections: 'Project' and 'Options'. In the 'Project' section, the 'Project Name' field is highlighted with a red box and contains the text 'STM32_Practice'. Below it, the 'Use default location' checkbox is checked. The 'Location' field shows the path 'C:/Users/fajar/STM32CubeIDE/workspace_1.13.1' with a 'Browse...' button next to it. The 'Options' section contains three sub-sections: 'Targeted Language' with radio buttons for 'C' (selected) and 'C++'; 'Targeted Binary Type' with radio buttons for 'Executable' (selected) and 'Static Library'; and 'Targeted Project Type' with radio buttons for 'STM32Cube' (selected) and 'Empty'. At the bottom of the window, there are four buttons: a help button (question mark icon), '< Back', 'Next >', and 'Finish'. The 'Finish' button is highlighted with a red box.

Mula-mula melakukan pengaturan pada Reset Clock Counter (RCC) dengan memilih High Speed External (HSE) berupa Crystal/Ceramic Resource seperti pada gambar berikut.

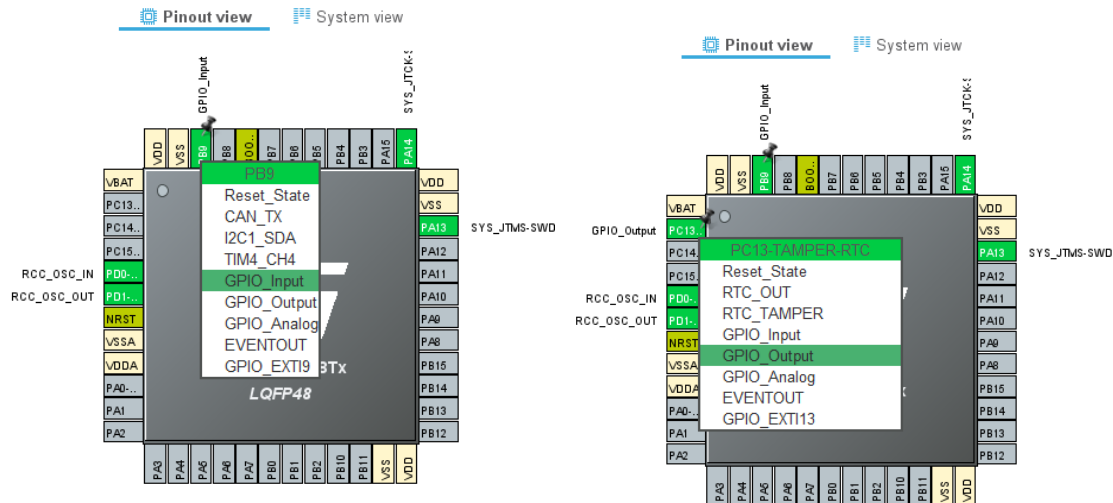


Pada bagian SYS, Debug dipilih Serial Wire dan pastikan Timebase Sourcenya merupakan SysTick

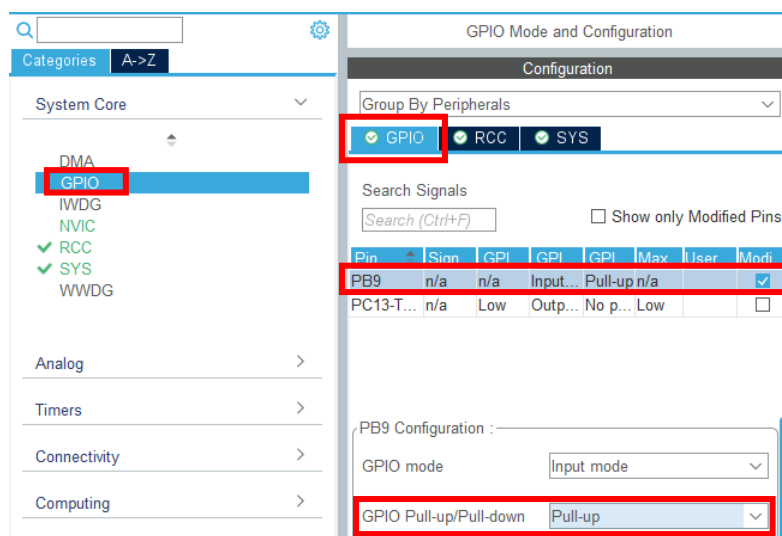


E. GPIO

Hampir semua pin pada Mikrokontroler STM32 berfungsi sebagai General Purpose Input Output (GPIO). Anda bisa mengatur sebuah pin sebagai GPIO_Input atau GPIO_Output. Sebagai contoh PB9 digunakan sebagai GPIO_Input dan PC13 sebagai GPIO_Output



Dikarenakan pin GPIO_Input tadi akan dihubungkan dengan Push Button, maka perlu dikonfigurasi jadi GPIO Pull Up sehingga user tidak perlu membuat rangkaian pull up resistor dengan push button.



Setelah semua sudah dikonfigurasi, selanjutnya dapat melakukan Save atau dengan command CTRL+S untuk men-generate main program berdasarkan yang telah kita konfigurasi.

Untuk program yang akan dijalankan pada main.c adalah sebagai berikut:

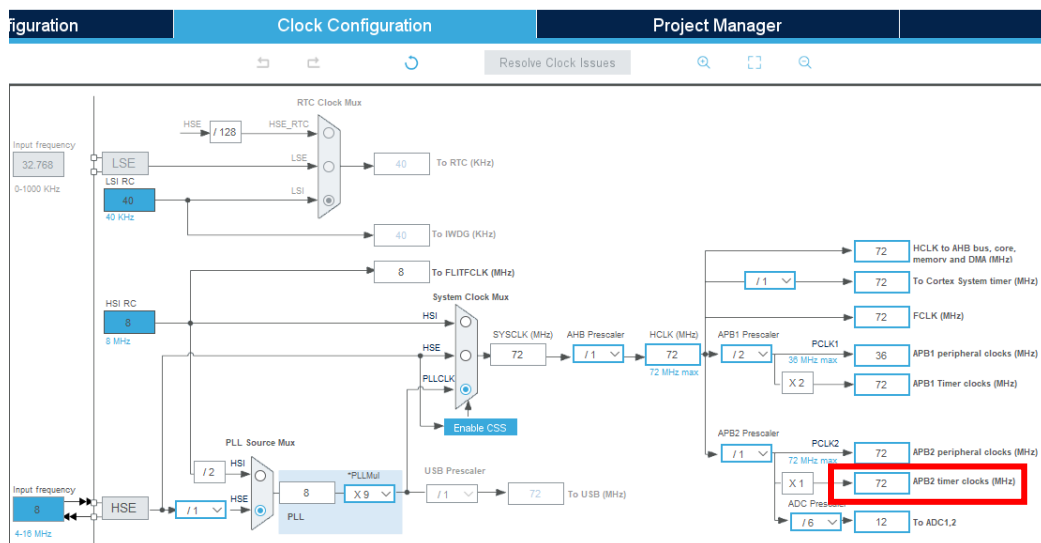
```
/* USER CODE BEGIN 2 */
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, SET); //beri state HIGH agar LED BuiltIn tidak menyala

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */

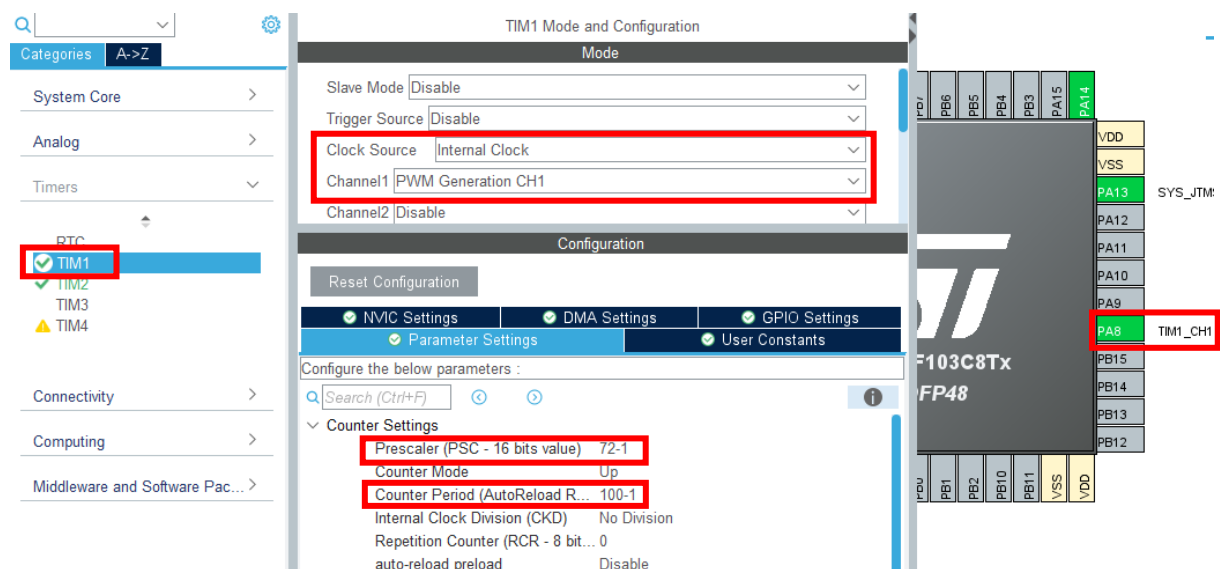
    /* USER CODE BEGIN 3 */
    pushButton = HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_9);
    if (pushButton == 0) HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, RESET);
    else HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, SET);
}
/* USER CODE END 3 */
```

F. Timer-PWM Output



Berikut merupakan CubeMX Setup nya:

Sesuai dengan datasheet, Timer 1 clock berhubungan dengan APB2 Timer clock dengan frekuensi 72MHz



Parameter setting tersebut akan mengatur frekuensi sinyal PWM dan resolusi sinyal PWM yang dibangkitkan dengan persamaan sebagai berikut.

$$\text{TIM CLOCK} = \frac{\text{APB TIM CLOCK}}{\text{PRESCALAR}}$$

$$\text{FREQUENCY} = \frac{\text{TIM CLOCK}}{\text{ARR}}$$

$$\text{DUTY \%} = \frac{\text{CCR}x}{\text{ARR}} \times 100$$

Pengaturan Prescaler Timer perlu dibuat nilainya dikurang 1 dikarenakan pada register prescaler melakukan penjumlahan +1 pada nilai yang ditetapkan pada CubeMX

14.4.11 TIM1 and TIM8 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to $f_{CK_PSC} / (PSC[15:0] + 1)$.

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in "reset mode").

Berikut merupakan contoh program pembangkitan sinyal PWM dengan duty cycle 50% (CCR = 50)

```

/* USER CODE BEGIN 2 */
// HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, SET);
HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
TIM1->CCR1 = 50;

/* USER CODE END 2 */

```

G. Timer- Encoder Mode

Configuration
Clock Configuration
Project Manager

Resolve Clock Issues

Untuk pemrograman dilakukan sebagai berikut

Main.c

```
/* USER CODE BEGIN 0 */
uint32_t counter = 0;

int16_t count = 0;

int16_t position = 0;

int speed = 0;

void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
    if(htim == &htim2){
        counter = __HAL_TIM_GET_COUNTER(htim);

        count = (int16_t)counter;

        position = count/4;
    }
}
/* USER CODE END 0 */

/* USER CODE BEGIN 2 */
// HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, SET); //be
HAL_TIM_Encoder_Start_IT(&htim2, TIM_CHANNEL_ALL);
// HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
// TIM1->CCR1 = 50;

/* USER CODE END 2 */
```

Stm32fXXX_it.c

```
/* Private user code -----
/* USER CODE BEGIN 0 */
extern int16_t position;
int16_t oldpos = 0;
extern int speed;
int indx = 0;
/* USER CODE END 0 */

void SysTick_Handler(void)
{
    /* USER CODE BEGIN SysTick_IRQn 0 */
    indx++;

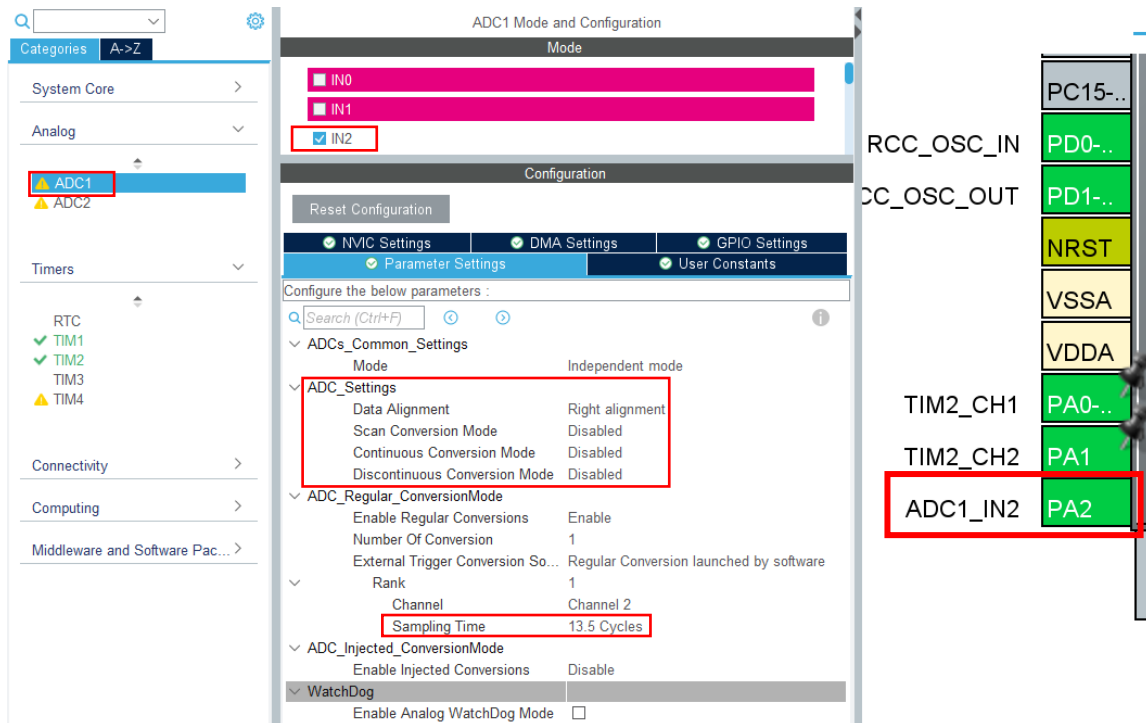
    if (indx == 500)
    {
        speed = ((position - oldpos)*2); // speed in clicks/sec
        oldpos = position;
        indx = 0;
    }

    /* USER CODE END SysTick_IRQn 0 */
    HAL_IncTick();
    /* USER CODE BEGIN SysTick_IRQn 1 */

    /* USER CODE END SysTick_IRQn 1 */
}
```

H. ADC

Ada 3 metode untuk melakukan ADC, yaitu metode Polling, metode Interrupt, dan metode DMA. Pada contoh berikut metode yang diterapkan ialah Polling yang merupakan metode yang paling mudah dari metode yang lain. Setup pada CubeMX ialah sebagai berikut



Untuk pemrograman ADC dengan metode polling ialah sebagai berikut

Main.c

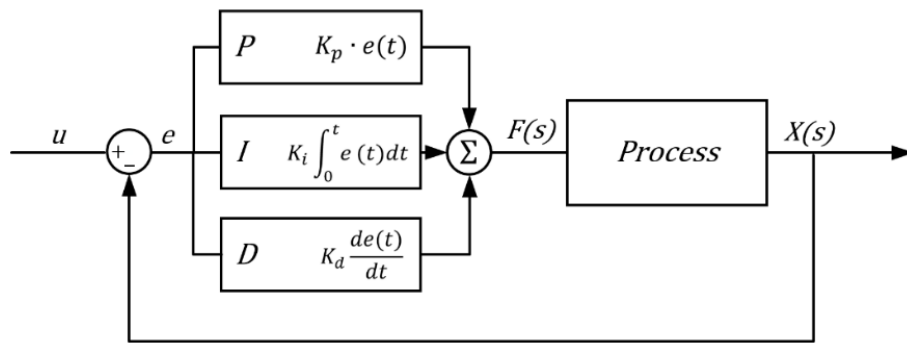
```
/* USER CODE BEGIN 0 */
uint32_t adc_val;
/* USER CODE END 0 */

/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, 1);
    adc_val = HAL_ADC_GetValue(&hadc1);
    HAL_ADC_Stop(&hadc1);

    HAL_Delay(1);
}
/* USER CODE END 3 */
}
```

I. PID Controller



Persamaan kontroler PID ini dapat dinyatakan sebagai berikut:

$$x(t) = K_p \cdot e(t) + K_i \cdot \int e(t) dt + K_d \cdot \frac{de(t)}{dt}$$

Keterangan:

K_p = konstanta proportional

K_i = konstanta integral

K_d = konstanta derivative

$e(t)$ = sinyal error

$x(t)$ = output kendali

Contoh program PID

```
uint32_t currentTimel = HAL_GetTick();
int deltaTimel = currentTimel - lastTimel;
if (deltaTimel >= T){
    errorl = spl - current_rpml;
    total_errorl += errorl;
    if (total_errorl >= max_control) total_errorl = max_control;
    else if (total_errorl <= min_control) total_errorl = min_control;
    int delta_errorl = errorl - last_errorl;
    motorSpeedl = kpl*errorl + (kil*T)*total_errorl + (kdl/T)*delta_errorl;
    if (motorSpeedl >= max_control) motorSpeedl = max_control;
    else if (motorSpeedl <= min_control) motorSpeedl = min_control;
    last_errorl = errorl;
    lastTimel = currentTimel;
}
```


Latihan Perancangan

1. Mengontrol LEDBuiltIn dengan Push Button
2. Menghasilkan PWM Signal
 - a. Menghasilkan sinyal PWM dengan Duty Cycle 75%, Frekuensi 10KHz
 - b. Menghasilkan sinyal PWM dengan Duty Cycle 50%, Frekuensi 8KHz
 - c. Menghasilkan sinyal PWM dengan Duty Cycle 30%, Frekuensi 5KHz
3. ADC dengan Potentiometer + LED Dimmer
4. Kendali Motor DC secara open loop dengan potentiometer
5. Encoder Motor
6. Kendali Motor Secara close loop
 - a. PID Controller Kecepatan
 - b. PID Controller Posisi